Porting Device Drivers from Windows 2000 (Win32) on IA32 to Windows NT64 (Win64) on Itanium™

(Assumption: A fully functional and working device driver already runs on IA32 Windows 2000)

Why do I have to make changes to the device drivers for Itanium™?

One of the main reasons behind this is the fact that on Win64 all pointers (Win64 logical addresses) in kernel mode are 64 bits. The user mode components can have 32bit pointers but these pointers cannot just be passed over to the kernel components. So, the user mode components may have different porting efforts depending on the memory size and performance requirements but the kernel mode components must be ported to handle 64 bits.

Other reasons are - data packing, data alignment and performance. Most drivers that directly talk to the hardware need to work with the page size or cache size that is hardware dependent. The data packing may need revisiting due to the pointer size changes. If the data is not aligned, there may be a loss in performance.

Other reasons that may have a bearing on this question are - does your device driver need anything special to support calls from native Win32 binaries OR do your drivers use any anything processor specific e.g. assembly instructions etc.

What are the steps to make it all happen?

Here are the steps to port a device driver written for Windows 2000 on IA32 to NT64 on Itanium™. Note that "code" here refers to device driver (kernel or user mode) and any configuration/utility/productivity application code that may accompany the device for a complete solution.

1. Get the IA-64 SDK and DDK from Intel – compilers, linkers, debuggers, libraries, IA64 Windows 2000, and Itanium™ simulator etc.
2. Prepare a IA32 Windows 2000 system that can be used as the reference system to help on troubleshooting problems on IA64 Windows 2000 since both Windows 2000 share the same code base
3. “Clean” the code for 64 bit – see below for challenges encountered in the process. If you have already done a Windows 2000 device driver *and* submitted the source code to Microsoft for inclusion in the Windows 2000 source tree, check with Microsoft if they have already "cleaned" the code for your device. Note that this "clean" code does not mean a "ready" driver - this may just be a good starting point
4. Make all the necessary changes needed to complete the driver port to work on IA32 and IA64
5. Regression test the driver on Windows 2000 (IA32)
6. Debug and test the 64 bit ready driver on the IA-64 simulator if possible
7. Get the Itanium™ Software Development Vehicle (SDV) – an Itanium™ based system with the operating system. Both the SoftSDV simulator and SDV system should be able to support the same IA64 Windows 2000 environment
8. Optimize the code on the SDV
9. Regression test the code on the SDV
10. Ship devices with production drivers and utilities at Itanium™ launch in 2H 2000

What are the typical challenges in porting device drivers to Windows NT64?

Following is a list of typical issues that a developer may have to consider during the port to Win64:

1. Invalid casts between pointers and `int`s (and/or `long`s). For example a pointer cast to an `int` will truncate the value.
2. Invalid `printf` specifiers. For example if a `%x` is used to print out the value of a pointer, the printed address will only be 8 digits. Instead use of `%08x` `%p` should be made.
3. Usage of existing APIs with pointers (and/or `long`s) as parameters and return values. For example APIs using a pointer to an `int` to pass in a string of arguments to a function. If the definition of the function calls for a pointer to be passed in via that argument then this will break for the 64 bit pointers.
4. Usage of undocumented/reserved bit fields
5. Use of hard coded values of size of data types
6. Use of hard coded bit shift values based on 32 bit size pointers
7. Use of hard coded constants in memory allocation function calls
8. Use of unguarded "`ifdefs`" defaulting to unwanted code generation. For example "`if defined (WIN32)`" may have to be changed to "`if defined (WIN32)||defined (_WIN64)`"
9. Use of constant offsets to access members of data structures. Data structure packing may be different based on the new architecture and compiler switches
10. Use of specific bits in pointer arithmetic based on a 32bit size. For example assuming that the sign bit is #31.
11. Use of inline assembly code. Itanium™ has a new instruction set which is not compatible with IA32 processors
12. Use of self-modifying code. Such a case may be using hard coded IA32 instruction opcodes which will not work on the Itanium™
13. Reading and writing of on-disk structures. Pointers residing on disk may not be wide enough to hold the expanded width
14. Data packing and alignment on IA-64 may be different - causing performance penalties
15. Use and manipulation of `CONTEXT` structure members will have to be modified since the registers are different on the Itanium™
16. Use of special/custom IOCTLs in the kernel mode driver that may deal with polymorphic data - they need to be able to handle requests from the Win32 as well as native Win64 subsystems
17. Assumption of the page size. On IA64 Windows 2000, the page size may be different

Need more information?

More information can be found at the following URLs:

1. **IA-64 Application Developer's Architecture Guide**
   A comprehensive description of the IA-64 application architecture, including the Application Instruction Set Architecture Guide, IA-32 Application Execution Model for IA-64, and IA-64 Performance Optimization Guide.
   Download now from http://developer.intel.com/design/ia64/downloads/adag.htm

2. **IA-64 Web Based Tutorials**
   The IA-64 internet-based tutorials provide the basic concepts of the IA-64 architecture and use of the new IA-64 instructions
   Learn now from http://developer.intel.com/vtune/cbts/ia64tuts/index.htm

3. **Getting Ready for 64-bit Windows**
   The team creating the 64-bit version of Microsoft® Windows® wants to make it possible for developers to use a single source-code base for their Win32®- and Win64™-based applications. The team has added support for this in the Platform SDK for Windows 2000. This overview describes how to make your source code support both 32-bit and 64-bit computing.
   View now at http://msdn.microsoft.com/library/psdk/buildapp/64bitwin_410z.htm

4. **Microsofts's* Win64 Porting Guide**
   How to optimize your code for Microsoft's Win64 operating system. Includes detailed guidelines. Presented by Oscar Newkerk from Microsoft at the February, 1999 Intel Developer Forum.
   Download now from http://developer.intel.com/design/ia64/downloads/ms.htm

5. **Designing 64-bit-Compatible Interfaces**
   Porting from the 32-bit Windows platform to the 64-bit Windows platform should not, by itself, create any problems for distributed applications, whether they use Remote Procedure Calls (RPC) directly or through DCOM. The RPC programming model specifies well-defined data sizes and integer types that are the same size on each end of the connection. Also, in the LLP64 abstract data model developed for 64-bit versions of the Microsoft® Windows® operating system, only the pointers expand to 64 bits—all other integer data types remain 32 bits.
   View now at http://msdn.microsoft.com/library/psdk/buildapp/mi-port64_9ius.htm

6. **Intel's Guide to Getting Started on IA-64 for Win64, Monterey, Linux, Modesto, HP-UX, Solaris, and Tru64**
   A short presentation on steps for porting applications to IA-64 on different operating systems
   View now at http://developer.intel.com/design/ia64/gettingstarted/

7. **IA-64 Architecture Innovations - Joint White paper between Intel & HP**
   Intel's IA-64 architecture is a unique combination of innovative features, which overcomes the performance limitations of traditional architectures. The IA-64 architecture is based on innovative techniques/features such as Explicit Parallelism, and Predication and Speculation, resulting in superior Instruction Level Parallelism (ILP) and increased instructions per cycle (IPC) to address the current and future requirements of these demanding Internet, high end server, and workstation applications. In addition, the IA-64 architecture provides headroom and scalability for continued future growth.
   Download now from http://developer.intel.com/design/ia64/downloads/ia64archinn.htm

8. **Intel's IA-64 Developer Site**
   For some of the above and other updated information on IA-64
   View now at http://developer.intel.com/design/IA64/